(19) (11) Publication number: 07175650 A

Generated Document.

PATENT ABSTRACTS OF JAPAN

(21) Application number: 05317876

(51) Intl. Cl.: G06F 9/32 G06F 9/32 G06F 9/38

(22) Application date: 17.12.93

(30) Priority:

(43) Date of application publication: 14.07.95

(84) Designated contracting states:

(71) Applicant: TOSHIBA CORP

(72) Inventor: MAEDA RIKA
TAKEUCHI YOICHIRO
SUZUKI SHINICHIRO
MORI YOSHIYA
TAKEUCHI KAZUAKI

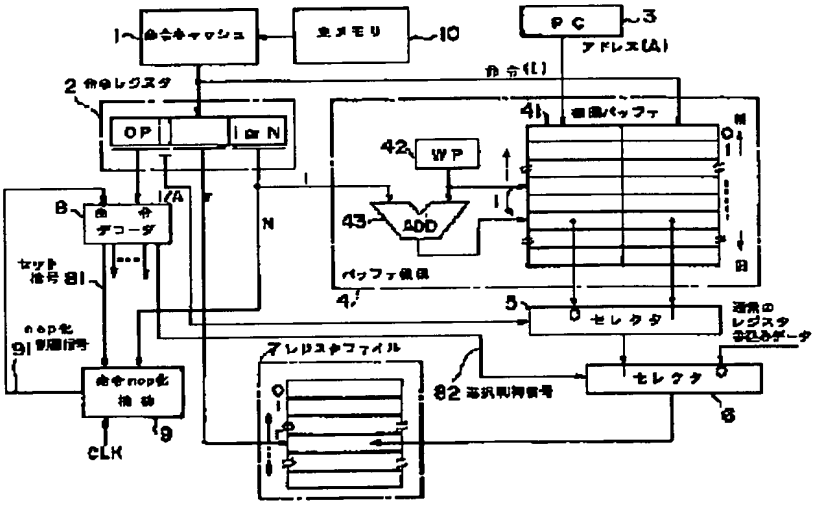
(74) Representative:

(54) ARITHMETIC PROCESSOR

(57) Abstract:

PURPOSE: To enable conditional processing with no branching, to easily obtain an instruction or an address before a designated cycle and to easily perform immediate data extraction or argument exchange.

CONSTITUTION: Each time an instruction is fetched into an instruction register 2, the instruction concerned and the address are successively stored in a circulation buffer 41. This fetched instruction is decoded by an instruction decoder 8. When this instruction is a SKIP(N) instruction, an instruction nop processing control signal 91 is outputted from an instruction nop processing mechanism 9 to the instruction decoder 8 just for the period of a cycle shown by a value N of the N field and during that period, the decode object instruction is processed as an nop instruction. When the instruction is a GETI(i, r) or GETA(i, r) instruction, among the instructions and addresses stored in the circulation buffer 41 before the cycle shown by a value (i) of the (i) field, the instruction or the address designated by the I/A bits is extracted to a register inside a register file 7 designated by the (r) field.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－175650

(43)公開日　平成７年(1995)７月14日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	9/32	３１０　Ｊ			
		３４０　Ｂ			
	9/38	３３０　Ｚ			

審査請求　未請求　請求項の数９　ＯＬ　（全 15 頁）

(21)出願番号　特願平5－317876

(22)出願日　平成５年(1993)12月17日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(72)発明者　前田　理香
東京都府中市東芝町１番地　株式会社東芝府中工場内

(72)発明者　竹内　陽一郎
東京都府中市東芝町１番地　株式会社東芝府中工場内

(72)発明者　鈴木　慎一郎
東京都府中市東芝町１番地　株式会社東芝府中工場内

(74)代理人　弁理士　鈴江　武彦

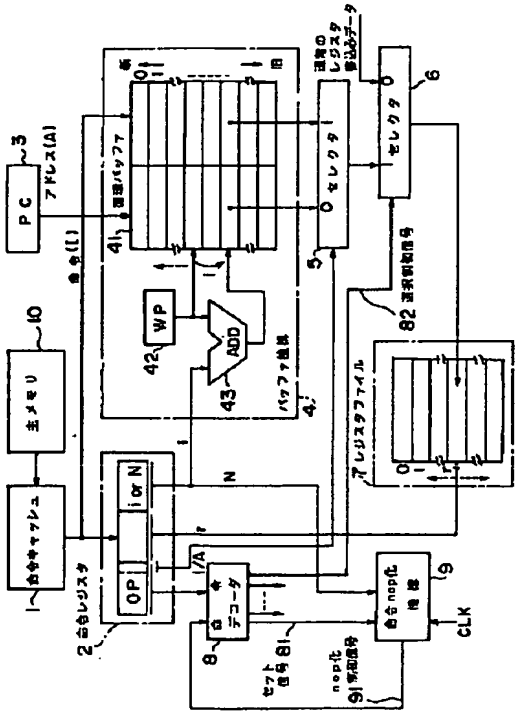
最終頁に続く

(54)【発明の名称】　演算処理装置

(57)【要約】

【目的】分岐なしの条件処理が図れ、指定サイクル前の命令またはアドレスが簡単に得られ、即値データ取出しや引数受け渡しが簡単に行えるようにする。

【構成】命令レジスタ２に命令がフェッチされる毎に、その命令とアドレスが循環バッファ４１に順次格納される。このフェッチされた命令は命令デコーダ８によりデコードされる。この命令がＳＫＩＰ（Ｎ）命令の場合、そのＮフィールドの値Ｎの示すサイクルの期間だけ、命令ｎｏｐ化機構９から命令デコーダ８に命令ｎｏｐ化制御信号９１が出力され、その期間中、デコード対象命令がｎｏｐ命令として扱われる。また、ＧＥＴＩ（ｉ，ｒ）またはＧＥＴＡ（ｉ，ｒ）命令の場合、そのｉフィールドの値ｉの示すサイクル前に循環バッファ４１に格納された命令とアドレスのうち、そのＩ／Ａビットの指定する命令またはアドレスが、そのｒフィールドの指定するレジスタファイル７内レジスタに取出される。



1

【特許請求の範囲】

【請求項１】命令群が記憶された記憶手段と、

前記記憶手段から取出された命令をデコードし、同命令を実行するのに必要な各種デコード信号を出力する命令デコード手段であって、当該命令が後続の命令を指定命令数だけノー・オペレーション化する指定命令数ノー・オペレーション化命令の場合に、対応する特定デコード信号を出力する命令デコード手段と、

前記命令デコード手段から前記特定デコード信号が出力された場合、当該命令デコード手段によってデコードされている前記指定命令数ノー・オペレーション化命令の指定する命令数に相当するサイクルの期間中、その間にデコードされる後続の命令をノー・オペレーション命令として扱うためのノー・オペレーション化制御信号を前記命令デコード手段に出力する命令ノー・オペレーション化手段とを具備し、前記命令デコード手段は、前記命令ノー・オペレーション化手段から前記ノー・オペレーション化制御信号が出力されている期間、デコード対象となる命令を、その命令種別に無関係にノー・オペレーション命令として扱うことを特徴とする演算処理装置。

【請求項２】前期記憶手段に、条件が成立したときのみ実行する命令群をノー・オペレーション化するための前記指定命令数ノー・オペレーション化命令を、この命令群の命令の直前に挿入して記憶しておき、当該ノー・オペレーション化命令が実行された場合には、当該ノー・オペレーション化命令で指定される数の命令がノーオペレーション化されるようにしたことを特徴とする請求項１記載の演算処理装置。

【請求項３】命令群が記憶された記憶手段と、

前記記憶手段から取出された命令をデコードし、同命令を実行するのに必要な各種デコード信号を出力する命令デコード手段であって、当該命令が後続の命令を指定命令数だけノー・オペレーション化する指定命令数ノー・オペレーション化命令の場合には、対応する第１のデコード信号を出力し、当該命令が指定サイクル前の命令を取出す命令履歴取出し命令の場合には、対応する第２のデコード信号を出力する命令デコード手段と、

前記デコード手段によるデコードの対象となる命令を順次格納するためのバッファ手段と、

このバッファ手段における前記命令の格納先を順次切換え指定する書込みポインタ手段と、

前記命令デコード手段から前記第１のデコード信号が出力された場合、当該命令デコード手段によってデコードされている前記指定命令数ノー・オペレーション化命令の指定する命令数に相当するサイクルの期間中、その間にデコードされる後続の命令をノー・オペレーション命令として扱うためのノー・オペレーション化制御信号を前記命令デコード手段に出力する命令ノー・オペレーション化手段と、

前記命令デコード手段から前記第２のデコード信号が出力されている場合、当該命令デコード手段によってデコードされている前記命令履歴取出し命令の指定するサイクル数と前記書込みポインタ手段の指す前記バッファ手段内位置をもとに、このバッファ手段から指定サイクル前の命令を取得する手段とを具備し、前記命令デコード手段は、前記命令ノー・オペレーション化手段から前記ノー・オペレーション化制御信号が出力されている期間、デコード対象となる命令を、その命令種別に無関係にノー・オペレーション命令として扱うことを特徴とする演算処理装置。

【請求項４】命令群が記憶された記憶手段と、

前記記憶手段から取出された命令をデコードし、同命令を実行するのに必要な各種デコード信号を出力する命令デコード手段であって、当該命令が後続の命令を指定命令数だけノー・オペレーション化する指定命令数ノー・オペレーション化命令の場合には、対応する第１のデコード信号を出力し、当該命令が指定サイクル前の命令のアドレスを取出すアドレス履歴取出し命令の場合には、対応する第２のデコード信号を出力する命令デコード手段と、

前記デコード手段によるデコードの対象となる命令のアドレスを順次格納するためのバッファ手段と、

このバッファ手段における前記アドレスの格納先を順次切換え指定する書込みポインタ手段と、

前記命令デコード手段から前記第１のデコード信号が出力された場合、当該命令デコード手段によってデコードされている前記指定命令数ノー・オペレーション化命令の指定する命令数に相当するサイクルの期間中、その間にデコードされる後続の命令をノー・オペレーション命令として扱うためのノー・オペレーション化制御信号を前記命令デコード手段に出力する命令ノー・オペレーション化手段と、

前記命令デコード手段から前記第２のデコード信号が出力されている場合、当該命令デコード手段によってデコードされている前記アドレス履歴取出し命令の指定するサイクル数と前記書込みポインタ手段の指す前記バッファ手段内位置をもとに、このバッファ手段から指定サイクル前の命令のアドレスを取得する手段とを具備し、前記命令デコード手段は、前記命令ノー・オペレーション化手段から前記ノー・オペレーション化制御信号が出力されている期間、デコード対象となる命令を、その命令種別に無関係にノー・オペレーション命令として扱うことを特徴とする演算処理装置。

【請求項５】命令群が記憶された記憶手段と、

前記記憶手段から取出された命令をデコードし、同命令を実行するのに必要な各種デコード信号を出力する命令デコード手段であって、当該命令が後続の命令を指定命令数だけノー・オペレーション化する指定命令数ノー・オペレーション化命令の場合には、対応する第１のデコード信号を出力し、当該命令が指定サイクル前の命令を

取出す命令履歴取出し命令または指定サイクル前の命令のアドレスを取出すアドレス履歴取出し命令の場合には、対応する第２のデコード信号を出力する命令デコード手段と、

前記デコード手段によるデコードの対象となる命令およびそのアドレスを順次格納するためのバッファ手段と、

このバッファ手段における前記命令およびアドレスの格納先を順次切換え指定する書込みポインタ手段と、

前記命令デコード手段から前記第１のデコード信号が出力された場合、当該命令デコード手段によってデコードされている前記指定命令数ノー・オペレーション化命令の指定する命令数に相当するサイクルの期間中、その間にデコードされる後続の命令をノー・オペレーション命令として扱うためのノー・オペレーション化制御信号を前記命令デコード手段に出力する命令ノー・オペレーション化手段と、

前記命令デコード手段から前記第２のデコード信号が出力されている場合、当該命令デコード手段によってデコードされている前記命令履歴取出し命令またはアドレス履歴取出し命令の指定するサイクル数と前記書込みポインタ手段の指す前記バッファ手段内位置をもとに、このバッファ手段から指定サイクル前の命令またはアドレスを取得する手段とを具備し、前記命令デコード手段は、前記命令ノー・オペレーション化手段から前記ノー・オペレーション化制御信号が出力されている期間、デコード対象となる命令を、その命令種別に無関係にノー・オペレーション命令として扱うことを特徴とする演算処理装置。

【請求項６】前期記憶手段に、前記指定命令数ノー・オペレーション命令、当該命令によりノー・オペレーション化される命令部分に設定される即値データ、および取得したい即値データを命令と見なしてサイクル数を指定することで前記バッファ手段から取出すための前記命令履歴取出し命令を記憶しておき、当該命令履歴取出し命令が実行された場合に、当該命令履歴取出し命令の指定するサイクル前の前記即値データを前記バッファ手段から取得するようにしたことを特徴とする請求項３または請求項５記載の演算処理装置。

【請求項７】前期記憶手段に、前記指定命令数ノー・オペレーション命令、当該命令によりノー・オペレーション化される命令部分に設定される手続き呼出しの引数、および取得したい引数を命令と見なしてサイクル数を指定することで前記バッファ手段から取出すための前記命令履歴取出し命令を記憶しておき、当該取出し命令が実行された場合に、当該取出し命令の指定するサイクル前の前記引数を前記バッファ手段から取得するようにしたことを特徴とする請求項３または請求項５記載の演算処理装置。

【請求項８】前期記憶手段内の、実行開始サイクルよりＮサイクル後にエラートラップの発生する可能性のある

命令が実際にエラートラップした場合の飛び先に、サイクル数Ｎ＋１を指定する前記アドレス履歴取出し命令を記憶しておき、当該取出し命令が実行された場合に、当該取出し命令の指定するサイクル前の前記エラートラップした命令のアドレスを前記バッファ手段から取得するようにしたことを特徴とする請求項４または請求項５記載の演算処理装置。

【請求項９】前記記憶手段内の、手続き呼出し命令の飛び先に、サイクル数１を指定する前記アドレス履歴取出し命令を記憶しておき、当該取出し命令が実行された場合に、当該取出し命令の指定するサイクル前の前記手続き命令のアドレスを前記バッファ手段から取得し、そのアドレスの次のアドレスを戻りアドレスとして取得することを特徴とする請求項４または請求項５記載の演算処理装置。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】この発明は、後続の命令を指定命令数だけｎｏｐ（ノー・オペレーション）化する専用命令を含む命令列を実行する演算処理装置に関する。

【０００２】

【従来の技術】一般に、演算処理装置で条件処理（条件分岐処理）を実行可能とするためには、分岐先の命令にラベルを付けておくのが一般的である。このような条件処理のための命令列について、図８を参照して説明する。

【０００３】まず、図８（ａ）は、ｎの値により異なる命令処理を実行するためのソースコードの一例を示すもので、ｎ＝０のときは命令＃１を、ｎ＝１のときは命令＃２を、そしてｎ＝２のときは命令＃３をそれぞれ実行して、例えば共通の命令処理に進む条件処理を指定している。ここでの条件処理は、例えばエラー要因ｎ（０～２）によって、その要因ｎ（０～２）別にトラップを発生させる命令＃１～＃３に分岐する場合であるものとする。

【０００４】さて、図８（ａ）に示すようなソースコードの指定する条件処理を、演算処理装置で実行可能とするためには、同ソースコードをコンパイルしてオブジェクトコード（ロードモジュール）を作成する必要がある。従来、条件処理は、条件分岐命令と分岐先の命令との組合せにより実現されることから、図８（ａ）に示すソースコードは、図８（ｂ）に示すようにコンパイルされるのが一般的である。図中ＳＵＢ（ｎ－０），ＳＵＢ（ｎ－１），ＳＵＢ（ｎ－２）は、それぞれ（ｎ－０），（ｎ－１），（ｎ－２）の減算を指定する減算命令（ＳＵＢ命令）、ＪＮＺ（Ｌ１），ＪＮＺ（Ｌ２），ＪＮＺ（Ｌ３）は、先行する命令（ＳＵＢ命令）の演算結果がゼロでない場合に、ラベルＬ１，Ｌ２，Ｌ３が付された命令（分岐先命令）にジャンプすることを示す条件分岐命令、ＪＰ（Ｌ１００）は、Ｌ１００の命令に無

条件でジャンプするジャンプ命令である。

【０００５】このように、従来の演算処理装置で条件処理を実現するには、分岐先の命令にラベルを付けておかねばならず、レジスタを用いた分岐先アドレスの計算等が必要となる他、条件（上記の例ではｎの値）が複数の場合には、別々に条件判定を行うことから、命令数が増加するといった問題があった。

【０００６】また、エラー処理の自動化を行うためのエラー情報解析処理などにおいては、指定サイクル前の実行命令を知りたい場合が発生する。ところが従来の演算処理装置では、途中に条件処理が存在すると目的の命令を見つけることは不可能であり、人がロードモジュールをたどって見つけなければならなかった。

【０００７】また従来は、即値データ（即値オペランドデータ）や引数は、レジスタからのデータロードにより得る必要があった。更に、戻りアドレスをリンクレジスタに設定しておく必要があった。

【０００８】

【発明が解決しようとする課題】上記したように従来の演算処理装置では、条件処理を実現するのに、分岐先の命令にラベルを付けておかねばならず、レジスタを用いた分岐先アドレスの計算等が必要となる他、命令数が増加するといった問題があった。

【０００９】また、指定サイクル前の実行命令を見つけるには、人がロードモジュールをたどらなければならないという問題もあった。また、レジスタからのデータロードによるメモリのオーバヘッドやレジスタを使用することによるレジスタ増に伴うコストの増加を招くという問題もあった。

【００１０】この発明は上記事情を考慮してなされたものでその目的は、分岐なしの条件処理が実現でき、もってレジスタを用いた分岐アドレスの計算等を不要とする他、命令数を削減できる演算処理装置を提供することにある。

【００１１】この発明の他の目的は、指定サイクル前の命令またはアドレスの少なくとも一方が簡単に取得できる演算処理装置を提供することにある。この発明の更に他の目的は、即値データの取出しや引数の受け渡しがレジスタからのロードを必要とせずに簡単に行える演算処理装置を提供することにある。この発明の更に他の目的は、手続き呼出し時の戻りアドレスが、リンクレジスタへの設定操作を必要とせずに取得できる演算処理装置を提供することにある。

【００１２】

【課題を解決するための手段および作用】この発明は、命令デコード手段によりデコードされている命令が後続の命令を指定命令数だけノー・オペレーション（ｎｏｐ）化する指定命令数ノー・オペレーション化命令（ＳＫＩＰ（Ｎ）命令）の場合に、当該命令デコード手段から特定デコード信号（第１のデコード信号）が出力される構成とすると共に、当該命令デコード手段から特定デコード信号（第１のデコード信号）が出力された場合、当該命令デコード手段によってデコードされているＳＫＩＰ（Ｎ）命令の指定する命令数Ｎに相当するＮサイクルの期間中、その間にデコードされる後続の命令をｎｏｐ命令として扱うためのｎｏｐ化制御信号を命令デコード手段に出力する命令ｎｏｐ化手段（命令ノー・オペレーション化手段）を設けたことを特徴とするものである。

【００１３】このような構成においては、命令デコード手段および命令ｎｏｐ化手段の組合せ動作により、ＳＫＩＰ（Ｎ）命令で指定された飛ばしたい個数だけ後続の命令がｎｏｐ化されるため、分岐なしで条件処理を実現することが可能となる。

【００１４】また、この発明は、上記命令デコード手段によりデコードされる命令およびそのアドレスを順次格納するためのバッファ手段と、このバッファ手段における上記命令およびアドレスの格納先を順次切換え指定する書込みポインタ手段とを更に設ける他、上記命令デコード手段によりデコードされている命令が、指定サイクル前の命令を取出す命令履歴取出し命令（ＧＥＴＩ（ｉ，ｒ）命令）または指定サイクル前の命令のアドレスを取出すアドレス履歴取出し命令（ＧＥＴＡ（ｉ，ｒ）命令）の場合には、当該命令デコード手段から第２のデコード信号が出力される構成とし、この第２のデコード信号が出力されている場合、命令デコード手段によってデコードされているＧＥＴＩ（ｉ，ｒ）命令またはＧＥＴＡ（ｉ，ｒ）命令の指定するサイクル数ｉと書込みポインタ手段の指すバッファ手段内位置をもとに、このバッファ手段からｉサイクル前の命令またはアドレスを取得するようにしたことを特徴とする。

【００１５】このような構成においては、命令デコード手段によりデコードされる命令およびそのアドレスがバッファ手段に順次格納される。そして、命令デコード手段によりＧＥＴＩ（ｉ，ｒ）またはＧＥＴＡ（ｉ，ｒ）命令がデコードされた場合には、バッファ手段に格納されている情報のうち、ＧＥＴＩ（ｉ，ｒ）またはＧＥＴＡ（ｉ，ｒ）命令中のｉフィールドの指定するサイクル（ｉサイクル）前の命令またはそのアドレスが、例えばＧＥＴＩ（ｉ，ｒ）またはＧＥＴＡ（ｉ，ｒ）命令中のｒフィールドの指定するレジスタファイル内レジスタに取出される。

【００１６】したがって、履歴を取得したい命令よりｉサイクル後に実行される位置にＧＥＴＩ（ｉ，ｒ）命令を用意しておくことで、当該命令ＧＥＴＩ（ｉ，ｒ）命令が実行された場合には、当該ＧＥＴＩ（ｉ，ｒ）命令中のｉフィールドの値で指定されたｉサイクル前の命令を命令履歴として、例えば当該ＧＥＴＩ（ｉ，ｒ）命令中のｒフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【００１７】同様に、ＳＫＩＰ（Ｎ）命令によりｎｏｐ化される命令部分に、即値データ（即値オペランドデータ）または手続き呼出しの固定引数を設定しておき、更にその後にＧＥＴＩ（ｉ，ｒ）命令を用意して、これらを実行させることにより、バッファ手段に格納されている情報のうち、当該ＧＥＴＩ（ｉ，ｒ）命令中のｉフィールドの値でｉサイクル前の命令として指定された即値データまたは固定引数を、例えば当該ＧＥＴＩ（ｉ，ｒ）命令中のｒフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【００１８】同様に、実行開始サイクルよりＮサイクル後にエラートラップの発生する可能性のある命令（対象命令）が実際にエラートラップした場合の飛び先に、サイクル数Ｎ＋１、即ちｉ＝Ｎ＋１を指定するＧＥＴＡ（ｉ，ｒ）命令を用意しておくことで、上記対象命令がエラートラップして当該ＧＥＴＡ（ｉ，ｒ）命令に実行が移った場合に、バッファ手段に格納されている情報のうち、当該ＧＥＴＡ（ｉ，ｒ）命令中のｉフィールドの値で指定されたＮ＋１サイクル前の命令のアドレス、即ち上記対象命令のアドレスをエラー発生命令のアドレスとして、例えば当該ＧＥＴＡ（ｉ，ｒ）命令中のｒフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【００１９】同様に、手続き呼出し命令の飛び先に、サイクル数１を指定するＧＥＴＡ（ｉ，ｒ）命令を用意しておくことで、当該ＧＥＴＡ（ｉ，ｒ）が呼出された場合に、バッファ手段に格納されている情報のうち、当該ＧＥＴＡ（ｉ，ｒ）命令中のｉフィールドの値で指定された１サイクル前の命令のアドレス、即ち上記手続き呼出し命令のアドレスを、戻りアドレスの１つ前のアドレスとして、例えば当該ＧＥＴＡ（ｉ，ｒ）命令中のｒフィールドの指定するレジスタファイル内レジスタに取出すことが可能となる。

【００２０】

【実施例】以下、この発明をパイプライン制御方式の演算処理装置に適用した一実施例につき、図面を参照して説明する。図１は同実施例に係る演算処理装置の主要部の構成を示すブロック図である。

【００２１】同図において、１は後述する主メモリ１０に格納されている命令群の一部の写しが置かれる命令キャッシュである。この命令キャッシュ１に実行すべき主メモリ１０上の命令が存在しないミスヒット時には、当該命令が主メモリ１０から命令キャッシュ１に読込まれて使用される。なお、命令キャッシュ１は、命令取出しの高速化のために設けられるもので、必ずしも必要ではない。

【００２２】２は命令キャッシュ１から取出された（フェッチされた）命令が保持される命令レジスタである。ここで、本実施例で適用される新規な命令について図２を参照して説明する。

【００２３】この新規な命令は、指定命令数ｎｏｐ化命令、命令履歴取出し命令およびアドレス履歴取出し命令の３種である。まず、指定命令数ｎｏｐ化命令は、後続の命令を指定命令数だけｎｏｐ化することを指定するもので、図２（ａ）に示すように指定命令数ｎｏｐ化命令であることを示すＯＰコード・フィールド（オペレーション・コード・フィールド）、およびｎｏｐ化する命令数Ｎを指定するフィールド（Ｎフィールド）とを有する。この指定命令数ｎｏｐ化命令をＳＫＩＰまたはＳＫＩＰ（Ｎ）のように表現する。

【００２４】次に命令履歴取出し命令は、指定サイクル前の命令を後述する循環バッファ４１から取出すことを指定するもので、図２（ｂ）に示すように、命令履歴取出し命令であることを示すＯＰコード・フィールド、取出した命令の格納先レジスタを指定するレジスタ（デスティネーション・レジスタ）指定フィールド（ｒフィールド）、および指定サイクル数ｉが設定される指定サイクル数フィールド（ｉフィールド）を有する。この命令履歴取出し命令をＧＥＴＩまたはＧＥＴＩ（ｉ，ｒ）のように表現する。なお、上記ｉフィールドの位置およびサイズは、指定命令数ｎｏｐ化命令（ＳＫＩＰ（Ｎ）命令）のＮフィールドのそれに一致する。

【００２５】次にアドレス履歴取出し命令は、指定サイクル前のアドレスを循環バッファ４１から取出すことを指定するもので、命令履歴取出し命令と同様の形式であり、図２（ｃ）に示すように、アドレス履歴取出し命令であることを示すＯＰコード・フィールド、取出したアドレスの格納先レジスタを指定するレジスタ（デスティネーション・レジスタ）指定フィールド（ｒフィールド）、および指定サイクル数ｉが設定される指定サイクル数フィールド（ｉフィールド）を有する。このアドレス履歴取出し命令をＧＥＴＡまたはＧＥＴＡ（ｉ，ｒ）のように表現する。

【００２６】ＧＥＴＩ（ｉ，ｒ）命令およびＧＥＴＡ（ｉ，ｒ）命令のＯＰコード・フィールドの例えば最下位ビットはＩ／Ａビットとして位置付けられている。ＧＥＴＩ（ｉ，ｒ）とＧＥＴＡ（ｉ，ｒ）の相違は、このＩ／Ａビットの値が異なる点であり、ＧＥＴＩ（ｉ，ｒ）のＩ／Ａビットは“１”、ＧＥＴＡ（ｉ，ｒ）のＩ／Ａビットは“０”である。この逆としても構わないことは勿論である。

【００２７】再び図１を参照すると、３は（命令キャッシュ１から）命令レジスタ２にフェッチされる命令の（主メモリ１０上の）アドレスを指定するプログラムカウンタ（ＰＣ）である。プログラムカウンタ３は、各命令フェッチサイクル毎に、インクリメントされる。但し、ジャンプ等が発生した場合には、プログラムカウンタ３の内容は、ジャンプ先のアドレスに更新される。

【００２８】４は命令キャッシュ１から命令レジスタ２に取出された命令と、その際のプログラムカウンタ３の

値（アドレス）の対を、各命令フェッチサイクル毎に順次保持するためのバッファ機構である。

【００２９】バッファ機構４は、各エントリがサイクリックに使用されるバッファ（循環バッファ）４１、この循環バッファ４１内の命令・アドレスの書込み先エントリを指す書込み先ポインタ（ＷＰ）４２、および加算器（ＡＤＤ）４３を有している。加算器４３は、後述する命令デコーダ８によりデコードされている命令がＧＥＴＩ（ｉ，ｒ）命令またはＧＥＴＡ（ｉ，ｒ）命令の場合に、その命令中のｉフィールドの示すサイクル数と書込み先ポインタ４２の示す値との加算を行い、循環バッファ４１内の読出し対象エントリの情報（読出しポインタ情報）を生成するのに用いられる。

【００３０】５はバッファ機構４内の循環バッファ４１から読出される命令およびアドレスのうちのいずれか一方を命令デコーダ８によりデコードされている命令のＯＰコード・フィールド中の最下位ビット（同命令がＧＥＴＩ（ｉ，ｒ）命令またはＧＥＴＡ（ｉ，ｒ）命令の場合にはＩ／Ａビット）に応じて選択するセレクタ、６はセレクタ５の出力情報および通常のレジスタ書込みデータのうちの一方を後述する選択制御信号８２に応じて選択するセレクタ、７はレジスタ群からなるレジスタファイルである。レジスタファイル７は、セレクタ６の出力情報を保持するのに用いられる。

【００３１】８は命令レジスタ２に保持された命令のＯＰコード・フィールドをデコードして、各部を制御する各種デコード信号（制御信号）を出力する命令デコーダである。この命令デコーダ８は、ＯＰコード・フィールドの内容がＳＫＩＰ（Ｎ）命令を示す場合には、同命令中のＮフィールドの値（指定命令数）を後述する命令ｎｏｐ化機構９内に設定するためのセット信号８１を出力し、ＧＥＴＩ（ｉ，ｒ）命令またはＧＥＴＡ（ｉ，ｒ）命令の場合には、セレクタ６に対する選択制御信号８２を出力するように構成されている。また命令デコーダ８は、命令ｎｏｐ化機構９から命令ｎｏｐ化制御信号９１が出力されている期間中は、デコード対象となる命令を、その命令種別に無関係にｎｏｐ命令として扱う。

【００３２】９は命令ｎｏｐ化機構である。この命令ｎｏｐ化機構９は、命令デコーダ８からセット信号８１が出力された場合に、同デコーダ８によりデコードされているＳＫＩＰ（Ｎ）命令中のＮフィールドの値Ｎを内部に設定し、その値Ｎの示すサイクルの期間（Ｎサイクルの期間）だけ、命令ｎｏｐ化制御信号９１を出力する。命令ｎｏｐ化機構９は、例えば、命令デコーダ８のデコード対象となっているＳＫＩＰ（Ｎ）命令中のＮフィールドの値Ｎをセット信号８１に応じてプリセットし、その値が「０」になるまでパイプライン・サイクルのクロックＣＬＫに応じてダウンカウントするカウンタと、このカウンタの値が「０」でない期間、アクティブな命令ｎｏｐ化制御信号９１を出力するゲート回路（いずれも図示せず）により構成される。

【００３３】１０は命令列からなる各種プログラム、データ等が格納される主メモリである。次に、図１の構成の基本動作について、図３のフローチャート、および図４乃至図７の基本動作説明図を参照して説明する。

【００３４】まず、プログラムカウンタ３の指定するアドレスの命令が命令キャッシュ１から取出され、命令レジスタ２に読込まれたものとする。同時に、バッファ機構４内の書込み先ポインタ４２が次の循環バッファ４１内エントリを指すように更新され、命令キャッシュ１から取出された命令とプログラムカウンタ３の指定するアドレスの対が、当該更新後の書込み先ポインタ４２の指定する循環バッファ４１内エントリに書込まれる（記録される）（ステップＳ１）。これにより、書込み先ポインタ４２は、命令レジスタ２にフェッチされた命令とそのアドレスを示すことになる。

【００３５】この様子を、命令キャッシュ１から命令レジスタ２にフェッチされた命令が、主メモリ１０上のアドレス“１０００”のＡＤＤ（加算）命令（ここではＡ＋Ｂを指定するＡＤＤ命令）である場合を例に、図４に示す。

【００３６】なお、書込み先ポインタ４２は、循環バッファ４１内エントリのアドレスが小さくなる方向に更新（デクリメント）されるものとする。また、書込み先ポインタ４２の値が「０」の状態で更新された場合、同ポインタ４２の更新後の値はエントリアドレスの最大値を指すものとする。

【００３７】命令レジスタ２にフェッチされた命令は、命令デコーダ８に導かれる。命令デコーダ８は、この命令のＯＰコード・フィールドの内容をデコードし、対応するデコード信号を出力する。

【００３８】ここで、命令デコーダ８によりデコードされている命令が指定命令数ｎｏｐ化命令、即ちＳＫＩＰ（Ｎ）命令の場合について説明する。この場合、命令デコーダ８からセット信号８１が出力される。このセット信号８１は、命令デコーダ８によりデコードされているＳＫＩＰ（Ｎ）中のＮフィールドの値と共に命令ｎｏｐ化機構９に導かれる。

【００３９】すると命令ｎｏｐ化機構９は、命令デコーダ８によりデコードされているＳＫＩＰ（Ｎ）中のＮフィールドの値「Ｎ」を、セット信号８１に応じて内部設定し、その値の示すサイクルの期間（Ｎサイクルの期間）だけ、アクティブな命令ｎｏｐ化制御信号９１を出力する。

【００４０】命令ｎｏｐ化機構９からの命令ｎｏｐ化制御信号９１は命令デコーダ８に導かれる。すると命令デコーダ８は、命令ｎｏｐ化制御信号９１がアクティブな期間（Ｎサイクルの期間）、命令レジスタ２を介して導かれる後続のＮ命令を、その命令種別に無関係にｎｏｐ命令として扱う。

【００４１】このように、図１の構成の演算処理装置では、命令デコーダ８にてデコードされている命令がＳＫＩＰ（Ｎ）命令である場合（ステップＳ２）、後続のＮ命令がｎｏｐ化される（ステップＳ３）。

【００４２】したがって、主メモリ１０上の或るアドレスに例えばＮ＝２のＳＫＩＰ（Ｎ）命令、即ちＳＫＩＰ（２）命令が存在する場合には、図５に示すように、ＳＫＩＰ（２）命令に続く２つの命令がｎｏｐ化されることになる。

【００４３】次に、命令デコーダ８によりデコードされている命令が命令履歴取出し命令、即ちＧＥＴＩ（ｉ，ｒ）命令の場合について説明する。この場合、命令デコーダ８によりデコードされているＧＥＴＩ（ｉ，ｒ）命令中のｉフィールドの値「ｉ」と、その時点の書込み先ポインタ４２の値とが、加算器４３により加算される。そして、加算器４３の加算結果（である読出しポインタ情報）の指定する循環バッファ４１内エントリに記録されている命令およびアドレスの対、即ち現在書込み先ポインタ４２の指定しているエントリ中の命令に対してｉサイクル前の命令およびアドレスの対が、同バッファ４１から読出される。

【００４４】循環バッファ４１から読出された命令およびアドレスはセレクタ５に導かれる。このセレクタ５には、命令デコーダ８によりデコードされているＧＥＴＩ（ｉ，ｒ）命令のＯＰコード・フィールド中のＩ／Ａビットも（選択制御信号として）導かれる。このＩ／Ａビットの値は、ＧＥＴＩ（ｉ，ｒ）命令では“１”である。

【００４５】セレクタ５は、この例のようにＩ／Ａ＝１の場合、循環バッファ４１から読出された命令およびアドレスのうちの命令をセレクタ６に選択出力する。一方、命令デコーダ８は、デコード対象となっている命令がこの例のようにＧＥＴＩ（ｉ，ｒ）命令の場合、論理“１”の選択制御信号８２を出力する。この選択制御信号８２はセレクタ６に導かれる。

【００４６】セレクタ６は、この例のように選択制御信号８２が“１”の場合、セレクタ５の選択出力情報（ここでは命令）を選択する。このように、図１の構成の演算処理装置では、命令デコーダ８にてデコードされている命令がＧＥＴＩ（ｉ，ｒ）命令である場合（ステップＳ４）、ｉサイクル前の命令およびそのアドレスの対が循環バッファ４１から取出され、そのうちの命令がセレクタ５，６を介してレジスタファイル７のｒレジスタに取出される（ステップＳ５）。

【００４７】この様子を、命令デコーダ８にてデコードされているＧＥＴＩ（ｉ，ｒ）命令のアドレスが“１０００”であり、ｉサイクル前の命令がＳＵＢ（減算）命令（ここではＣ－Ｄを指定するＳＵＢ命令）である場合を例に、図６に示す。

【００４８】次に、命令デコーダ８によりデコードされている命令がアドレス履歴取出し命令、即ちＧＥＴＡ（ｉ，ｒ）命令の場合について説明する。この場合、命令デコーダ８によりデコードされているＧＥＴＡ（ｉ，ｒ）命令中のｉフィールドの値「ｉ」と、その時点の書込み先ポインタ４２の値とが、加算器４３により加算される。そして、加算器４３の加算結果（である読出しポインタ情報）の指定する循環バッファ４１内エントリに記録されている命令およびアドレスの対が、即ち現在書込み先ポインタ４２の指定しているエントリ中の命令に対してｉサイクル前の命令およびそのアドレスの対が、同バッファ４１から読出される。

【００４９】循環バッファ４１から読出された命令およびアドレスはセレクタ５に導かれる。このセレクタ５には、命令デコーダ８によりデコードされているＧＥＴＡ（ｉ，ｒ）命令のＯＰコード・フィールド中のＩ／Ａビットも（選択制御信号として）導かれる。このＩ／Ａビットの値は、ＧＥＴＡ（ｉ，ｒ）命令では“０”である。

【００５０】セレクタ５は、この例のようにＩ／Ａ＝０の場合、循環バッファ４１から読出された命令およびアドレスのうちのアドレスをセレクタ６に選択出力する。一方、命令デコーダ８は、デコード対象となっている命令がこの例のようにＧＥＴＡ（ｉ，ｒ）命令の場合、上記したＧＥＴＩ（ｉ，ｒ）命令のときと同様に論理“１”の選択制御信号８２を出力する。

【００５１】セレクタ６は、この例のように選択制御信号８２が“１”の場合、セレクタ５の選択出力情報（ここではアドレス）を選択する。セレクタ６により選択された情報（アドレス）は、この例のように、命令デコーダ８によりＧＥＴＡ（ｉ，ｒ）命令がデコードされている場合には、同命令中のｒフィールドの内容「ｒ」の指定するレジスタファイル７内レジスタ（ｒレジスタ）に格納される。

【００５２】このように、図１の構成の演算処理装置では、命令デコーダ８にてデコードされている命令がＧＥＴＡ（ｉ，ｒ）命令である場合（ステップＳ６）、ｉサイクル前の命令およびそのアドレスの対が循環バッファ４１から取出され、そのうちのアドレスがセレクタ５，６を介してレジスタファイル７のｒジスタに取出される（ステップＳ７）。

【００５３】この様子を、命令デコーダ８にてデコードされているＧＥＴＡ（ｉ，ｒ）命令のアドレスが“１０５０”であり、ｉサイクル前の命令のアドレスが“１０００”である場合を例に、図７に示す。

【００５４】次に、図１の演算処理装置において、ＳＫＩＰ命令（指定命令数ｎｏｐ化命令）を利用して、図８（ａ）に示したようなソースコードの指定する条件処理を、図８（ｃ）に示すように分岐なしで実現した例につき、説明する。なお、図８（ａ）に示すソースコードは、［従来の技術］の欄で説明したように、ｎ＝０のと

きは命令＃１を、ｎ＝１のときは命令＃２を、そしてｎ＝２のときは命令＃３をそれぞれ実行して、例えば共通の命令処理に進む条件分岐処理を指定するものである。

【００５５】さて、図８（ａ）に示すようなソースコードの指定する条件処理を、図１の構成の演算処理装置で実行可能とするためには、同ソースコードをコンパイルしてオブジェクトコード（ロードモジュール）を作成する必要がある。この際、条件が成立したときに実行する命令の前にＳＫＩＰ命令を挿入してコンパイルする。本実施例では、図８（ａ）に示すソースコードをコンパイルした場合、図８（ｃ）に示すように、ＳＫＩＰ（ｎ＊２）、命令＃１、ＳＫＩＰ（３）、命令＃２、ＳＫＩＰ（１）および命令＃３の順で続く命令列が作成される。

【００５６】前記したように、図１の構成の演算処理装置においては、ＳＫＩＰ（Ｎ）命令により、後続のＮ命令がｎｏｐ化される（図３ステップＳ２，Ｓ３）。したがって、図８（ｃ）に示した命令列の例では、ｎ＝０の場合には、ＳＫＩＰ（ｎ＊２）、即ちＳＫＩＰ（０）に続いて、命令＃１、ＳＫＩＰ（３）が順次実行され、更に後続の３命令（命令＃２、ＳＫＩＰ（１）、命令＃３）がｎｏｐ化された後、命令＃３の次の命令が実行される。

【００５７】同様に、ｎ＝１の場合には、ＳＫＩＰ（ｎ＊２）、即ちＳＫＩＰ（２）が実行されて、後続の２命令（命令＃１、ＳＫＩＰ（３））がｎｏｐ化された後、命令＃２、ＳＫＩＰ（１）が順次実行され、更に後続の１命令（命令＃３）がｎｏｐ化された後、命令＃３の次の命令が実行される。

【００５８】同様に、ｎ＝２の場合には、ＳＫＩＰ（ｎ＊２）、即ちＳＫＩＰ（４）が実行されて、後続の４命令（命令＃１、ＳＫＩＰ（３）、命令＃２、ＳＫＩＰ（１））がｎｏｐ化された後、ＳＫＩＰ（１）の次の命令＃３が実行され、更に次の当該命令＃３の次の命令が実行される。

【００５９】このように、図１に示す構成の演算処理装置では、ＳＫＩＰ命令（指定命令数ｎｏｐ化命令）を利用することで、分岐を用いずに条件処理を実現することができる。したがって、図８（ｃ）から明らかなように、条件処理のための命令列の構成命令数が、図８（ｂ）に示した従来の場合の命令列に比較して削減できる。また、命令数が削減できることから、処理の高速化が図れ、更に分岐発生に伴うパイプライン停止の問題がないため、一層の高速化が図れる。

【００６０】次に、図１の演算処理装置において、ＧＥＴＩ命令（命令履歴取出し命令）を利用して、指定サイクル前の命令を取得する場合（命令履歴取出し）につき、図９を参照して説明する。

【００６１】まず、ＧＥＴＩ命令を利用して、指定サイクル前の命令を取得できるようにするには、途中に分岐がないようにしておく必要がある。このため、途中で条件処理を必要とする場合には、先に述べたように、条件処理についてＳＫＩＰ命令を用いた形にコンパイルすることで、分岐なしの条件処理を実現できるようにしておく。

【００６２】図９は、主メモリ１０上に、ＳＫＩＰ（１）、何らかの１命令、命令＃１、何らかの２命令、ＧＥＴＩ（３，ｒ）の命令列が格納されている場合に、当該命令列が１命令ずつ順次実行され、ＧＥＴＩ（３，ｒ）が実行された際の様子を説明するためのものである。

【００６３】図９の例では、ＳＫＩＰ（１）、何らかの１命令、命令＃１、何らかの２命令、ＧＥＴＩ（３，ｒ）がＳＫＩＰ（１）から順に１命令ずつ命令レジスタ２にフェッチされる毎に、書込み先ポインタ４２がデクリメントされて、そのデクリメント後の当該ポインタ４２の指定する循環バッファ４１内エントリに、上記フェッチされた命令とそのアドレスが記録される（ステップＳ１）。また、ＳＫＩＰ（１）の次の１命令は、当該ＳＫＩＰ（１）の指定によりｎｏｐ化される（ステップＳ２，Ｓ３）。

【００６４】さて、ＧＥＴＩ（３，ｒ）が命令レジスタ２にフェッチされて、当該ＧＥＴＩ（３，ｒ）とそのアドレスが、図９に示すように、書込み先ポインタ４２の指す循環バッファ４１内エントリに記録され（ステップＳ１）、更に当該ＧＥＴＩ（３，ｒ）が命令デコーダ８でデコードされたものとする（ステップＳ４）。

【００６５】この場合、図６を参照して説明したことからも明らかなように、ＧＥＴＩ（３，ｒ）のｉフィールドの値「３」（ｉ＝３）と書込み先ポインタ４２の値との（加算器４３による）加算値で指定される循環バッファ４１内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ４２の指しているエントリ中の命令ＧＥＴＩ（３，ｒ）に対して３サイクル前の命令およびアドレスの対のうちの命令（ここでは命令＃１）が、図９に示すように、当該ＧＥＴＩ（３，ｒ）のｒフィールドの指定する、レジスタファイル７内のｒレジスタに格納される（ステップＳ５）。

【００６６】ここで、ＳＫＩＰ（１）を用いることで条件処理を分岐なしで実現していることから、途中に条件処理があっても、ＧＥＴＩ（３，ｒ）の実行により、指定サイクル前（ここでは３サイクル前）の命令（命令＃１）を、正しくレジスタファイル７内の指定レジスタ（ｒレジスタ）に取出すことができる。

【００６７】次に、図１の演算処理装置において、ＳＫＩＰ命令（指定命令数ｎｏｐ化命令）でｎｏｐ化された命令部分（位置）に即値データを格納しておき、当該即値データをＧＥＴＩ命令（命令履歴取出し命令）を利用してレジスタファイル７内に取出す場合（即値オペランド指定）につき、図１０を参照して説明する。

【００６８】まず、ＧＥＴＩ命令を利用して即値データ

を取出せるようにするには、コンパイルの際に、ＳＫＩＰ命令によりｎｏｐ化される命令位置に即値データを格納しておく必要がある。

【００６９】図１０は、主メモリ１０上に、ＳＫＩＰ（２）、定数＃１、定数＃２、ＧＥＴＩ（２，ｒ）の命令列（定数＃１，＃２も命令として見なす）が格納されている場合に、ＳＫＩＰ（２）から順に１命令ずつ実行され、ＧＥＴＩ（２，ｒ）が実行された際の様子を説明するためのものである。この図１０の例では、ＳＫＩＰ（２）でｎｏｐ化される２つの命令部分に、それぞれ定数＃１と定数＃２の２つの即値データが予め格納されている。

【００７０】図１０の例では、ＳＫＩＰ（２）、定数＃１、定数＃２、ＧＥＴＩ（２，ｒ）がＳＫＩＰ（２）から順に１命令ずつ命令レジスタ２にフェッチされる毎に、書込み先ポインタ４２がデクリメントされて、そのデクリメント後の当該ポインタ４２の指定する循環バッファ４１内エントリに、上記フェッチされた命令とそのアドレスが記録される（ステップＳ１）。また、ＳＫＩＰ（２）の次の２つの定数＃１，＃２は、当該ＳＫＩＰ（２）の指定によりｎｏｐ化される（ステップＳ２，Ｓ３）。したがって、ＳＫＩＰ（２）の次の２命令部分に、即値データである定数＃１，＃２が格納（設定）されていても、何ら問題はない。

【００７１】さて、ＧＥＴＩ（２，ｒ）が命令レジスタ２にフェッチされて、当該ＧＥＴＩ（２，ｒ）とそのアドレスが、図１０に示すように、書込み先ポインタ４２の指す循環バッファ４１内エントリに記録され（ステップＳ１）、更に当該ＧＥＴＩ（２，ｒ）が命令デコーダ８でデコードされたものとする（ステップＳ４）。

【００７２】この場合、図６を参照して説明したことからも明らかなように、ＧＥＴＩ（２，ｒ）のｉフィールドの値「２」（ｉ＝２）と書込み先ポインタ４２の値との（加算器４３による）加算値で指定される循環バッファ４１内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ４２の指しているエントリ中の命令ＧＥＴＩ（２，ｒ）に対して２サイクル前の命令およびアドレスの対のうちの命令（ここでは、命令に代えて格納されている定数＃１）が、図１０に示すように、当該ＧＥＴＩ（２，ｒ）のｒフィールドの指定する、レジスタファイル７内のｒレジスタに格納される（ステップＳ５）。

【００７３】このように、ＳＫＩＰ（２）によりｎｏｐ化される２命令部分に即値データである定数＃１，＃２を格納しておいた場合、定数＃２の次の位置のＧＥＴＩ（２，ｒ）の実行により、定数＃１（即値データ）を（レジスタからのデータロードを指定する）ロード命令を行うことなく、取得できる。なお、ＧＥＴＩ（２，ｒ）に代えてＧＥＴＩ（１，ｒ）を用いるならば、定数＃２を取得できることは勿論である。

【００７４】次に、図１の演算処理装置において、ＳＫＩＰ命令（指定命令数ｎｏｐ化命令）でｎｏｐ化された命令部分（位置）に引数を格納しておき、当該引数をＧＥＴＩ命令（命令履歴取出し命令）を利用してレジスタファイル７内に取出す場合（固定引数引渡し）につき、図１１を参照して説明する。

【００７５】まず、ＧＥＴＩ命令を利用して、引数（固定引数）を引渡せるようにするには、コンパイルの際に、ＳＫＩＰ命令によりｎｏｐ化される命令位置に関数（手続き呼出し）の引数を格納し、更に飛び先にＧＥＴＩ命令を挿入しておく必要がある。

【００７６】図１１は、主メモリ１０上に、ＳＫＩＰ（２）、引数ａｒｇ１、引数ａｒｇ２、手続き呼出し命令であるＣＡＬＬ（ＦＮ１）の命令列（引数ａｒｇ１，ａｒｇ２も命令として見なす）が格納され、更にＣＡＬＬ（ＦＮ１）の指定する飛び先ＦＮ１（に対応するアドレス）にＧＥＴＩ（２，ｒ）が格納されている場合に、ＳＫＩＰ（２）から順に１命令ずつ実行され、ＣＡＬＬ（ＦＮ１）の実行の後、その飛び先のＧＥＴＩ（２，ｒ）が実行された際の様子を説明するためのものである。この図１１の例では、ＳＫＩＰ（２）でｎｏｐ化される２つの命令部分に、それぞれ固定の引数ａｒｇ１と引数ａｒｇ２が予め格納されている。

【００７７】図１１の例では、ＳＫＩＰ（２）、引数ａｒｇ１、引数ａｒｇ２、ＣＡＬＬ（ＦＮ１）、ＧＥＴＩ（２，ｒ）がＳＫＩＰ（２）から順に１命令ずつ命令レジスタ２にフェッチされる毎に、書込み先ポインタ４２がデクリメントされて、そのデクリメント後の当該ポインタ４２の指定する循環バッファ４１内エントリに、上記フェッチされた命令とそのアドレスが記録される（ステップＳ１）。また、ＳＫＩＰ（２）の次の２つの引数ａｒｇ１，ａｒｇ２は、当該ＳＫＩＰ（２）の指定によりｎｏｐ化される（ステップＳ２，Ｓ３）。したがって、ＳＫＩＰ（２）の次の２命令部分に、引数ａｒｇ１，ａｒｇ２が格納（設定）されていても、何ら問題はない。

【００７８】さて、ＧＥＴＩ（２，ｒ）が命令レジスタ２にフェッチされて、当該ＧＥＴＩ（２，ｒ）とそのアドレスが、図１１に示すように、書込み先ポインタ４２の指す循環バッファ４１内エントリに記録され（ステップＳ１）、更に当該ＧＥＴＩ（２，ｒ）が命令デコーダ８でデコードされたものとする（ステップＳ４）。

【００７９】この場合、図６を参照して説明したことからも明らかなように、ＧＥＴＩ（２，ｒ）のｉフィールドの値「２」（ｉ＝２）と書込み先ポインタ４２の値との（加算器４３による）加算値で指定される循環バッファ４１内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ４２の指しているエントリ中の命令ＧＥＴＩ（２，ｒ）に対して２サイクル前の命令およびアドレスの対のうちの命令（ここでは、命令

に代えて格納されている引数ａｒｇ２）が、図１１に示すように、当該ＧＥＴＩ（２，ｒ）のｒフィールドの指定する、レジスタファイル７内のｒレジスタに格納される（ステップＳ５）。

【００８０】このように、ＳＫＩＰ（２）によりｎｏｐ化される２命令部分に関数（手続き呼出し）の引数ａｒｇ１，ａｒｇ２を格納し、飛び先にＧＥＴＩ（２，ｒ）を挿入しておいた場合、ＧＥＴＩ（２，ｒ）の実行により、引数ａｒｇ２を（レジスタからのデータロードを指定する）ロード命令を行うことなく、取得できる。なお、ＧＥＴＩ（２，ｒ）に代えてＧＥＴＩ（３，ｒ）を用いるならば、引数ａｒｇ１を取得できることは勿論である。

【００８１】ところで、スーパーコンピュータなどでは、１つの命令の実行完了を待たずに次の命令を実行可能な構成となっていることが多い。このようなものにおいては、エラートラップが発生しても、そのエラー発生命令のアドレスを取得することは困難である。そこで、図１の演算処理装置において、エラートラップが発生したら、ＧＥＴＡ命令（アドレス履歴取出し命令）に飛ぶようにしておき、当該ＧＥＴＡ命令を利用してエラー発生命令のアドレスをレジスタファイル７内に取出す場合（エラー発生命令のアドレス取出し）につき、図１２を参照して説明する。

【００８２】まず、ＧＥＴＡ命令を利用して、エラー発生命令のアドレスを取出せるようにするには、コンパイルの際に、エラートラップしたらＧＥＴＡ命令に飛ぶようにしておく必要がある。また、条件処理を必要とする場合には、ＳＫＩＰ命令を利用して分岐なしで条件処理が行えるようにしておく。

【００８３】図１２は、主メモリ１０上に、例えば３サイクルで実行結果が生成されるパイプライン演算命令である浮動小数点加算命令ｆａｄｄ と、後続する何らかの２命令とが格納され、更に当該２命令のうちの後の命令の実行中に先行する命令ｆａｄｄがエラートラップした場合の割込み先にＧＥＴＡ（３，ｒ）が格納されている場合に、命令ｆａｄｄ から順に１命令ずつ実行が開始され、命令ｆａｄｄ がエラートラップした結果、ＧＥＴＡ（３，ｒ）が実行された際の様子を説明するためのものである。なお、命令ｆａｄｄ のアドレスは“１０００”であるものとする。

【００８４】図１２の例では、命令ｆａｄｄ 、後続の２命令が１命令ずつ命令レジスタ２に順次フェッチされる毎に、書込み先ポインタ４２がデクリメントされて、そのデクリメント後の当該ポインタ４２の指定する循環バッファ４１内エントリに、上記フェッチされた命令とそのアドレスが記録される（ステップＳ１）。

【００８５】ここで、命令ｆａｄｄ が、実行開始サイクルより２サイクル後にエラートラップしたものとする。この場合、割込み先のＧＥＴＡ（３，ｒ）が命令レジスタ２にフェッチされると同時に、当該命令ｆａｄｄ とそのアドレスが、当該命令ｆａｄｄ の１サイクル前にフェッチされた命令（命令ｆａｄｄ に後続する２命令のうちの後の命令）の次に位置する循環バッファ４１内エントリに記録される（ステップＳ１）。そして、命令レジスタ２にフェッチされたＧＥＴＡ（３，ｒ）が命令デコーダ８でデコードされたものとする（ステップＳ６）。

【００８６】この場合、図７を参照して説明したことからも明らかなように、ＧＥＴＡ（３，ｒ）のｉフィールドの値「３」（ｉ＝３）と書込み先ポインタ４２の値との（加算器４３による）加算値で指定される循環バッファ４１内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ４２の指しているエントリ中の命令ＧＥＴＡ（３，ｒ）に対して３サイクル前のエラー発生命令ｆａｄｄおよびアドレスの対のうちのアドレス（ここでは“１０００”）が、図１２に示すように、当該ＧＥＴＩ（３，ｒ）のｒフィールドの指定する、レジスタファイル７内のｒレジスタに取出される（ステップＳ７）。

【００８７】次に、図１の演算処理装置において、ＣＡＬＬ命令（手続き呼出し命令）の飛び先にＧＥＴＡ命令を挿入すると共に、手続き終了位置にＲＥＴ命令（リターン命令）を挿入しておき、ＧＥＴＡ命令を利用してＣＡＬＬ命令のアドレスをレジスタファイル７内に取出す場合（手続き呼出し時の戻りアドレス指定）につき、図１３を参照して説明する。

【００８８】図１３は、主メモリ１０上に、ＣＡＬＬ（ＦＮ１）、当該ＣＡＬＬ（ＦＮ１）による手続き呼出しから戻った際に実行される命令＃１を先頭とする命令列が格納され、更にＣＡＬＬ（ＦＮ１）の指定する飛び先ＦＮ１（に対応するアドレス）から始まる領域にＧＥＴＡ（１，ｒ）を先頭命令とし、レジスタファイル７内のｒレジスタの内容に１を加えたアドレス（戻りアドレス）に実行を移すためのＲＥＴ（ｒ＋１）を最終命令とする命令列が格納されている場合に、ＣＡＬＬ（ＦＮ１）から順に１命令ずつ実行が開始され、ＲＥＴ（ｒ＋１）の実行により、戻り先の命令＃１にリターンした際の様子を説明するためのものである。なお、ＣＡＬＬ（ＦＮ１），命令＃１のアドレスは、それぞれ“１０００”，“１００１”であるものとする。

【００８９】図１３の例では、まずＣＡＬＬ（ＦＮ１）が、続いてＧＥＴＡ（１，ｒ）が命令レジスタ２にフェッチされる。この命令フェッチの都度、書込み先ポインタ４２がデクリメントされて、そのデクリメント後の当該ポインタ４２の指定する循環バッファ４１内エントリに、上記フェッチされた命令とそのアドレスが記録される（ステップＳ１）。

【００９０】ここで、命令レジスタ２にフェッチされたＧＥＴＡ（１，ｒ）が、命令デコーダ８でデコードされると（ステップＳ６）、図７を参照して説明したことか

らも明らかなように、ＧＥＴＡ（１，ｒ）のｉフィールドの値「１」（ｉ＝１）と書込み先ポインタ４２の値との（加算器４３による）加算値で指定される循環バッファ４１内エントリから読出される命令およびアドレスの対、即ち現在書込み先ポインタ４２の指しているエントリ中の命令ＧＥＴＡ（１，ｒ）に対して１サイクル前の命令およびアドレスの対のうちのアドレス（ここでは、ＣＡＬＬ（ＦＮ１）のアドレス“１０００”）が、図１３に示すように、当該ＧＥＴＡ（１，ｒ）のｒフィールドの指定する、レジスタファイル７内のｒレジスタに格納される（ステップＳ７）。

【００９１】さて、ＧＥＴＡ（１，ｒ）に後続する命令列が１命令ずつ順次命令レジスタ２にフェッチされ、やがてＲＥＴ（ｒ＋１）が当該命令レジスタ２にフェッチされたものとする。すると、ＲＥＴ（ｒ＋１）とそのアドレスが、書込み先ポインタ４２の指定する循環バッファ４１内エントリに記録される。

【００９２】そして、この命令レジスタ２にフェッチされたＲＥＴ（ｒ＋１）が命令デコーダ８でデコードされて、そのデコード結果に従って実行される（ステップＳ６，Ｓ８）。このＲＥＴ（ｒ＋１）は、レジスタファイル７内のｒレジスタの内容に１を加えたアドレスを戻りアドレスとして、当該アドレスに実行を移すリターン命令である。ここで、レジスタファイル７内のｒレジスタには、先のＧＥＴＡ（１，ｒ）の実行により、ＣＡＬＬ（ＦＮ１）のアドレス“１０００”が格納されている。

【００９３】したがって、ＲＥＴ（ｒ＋１）の実行により、アドレス“１００１”の命令、即ちＣＡＬＬ（ＦＮ１）の次のアドレスの命令＃１にリターンすることになる。具体的には、アドレス“１００１”がプログラムカウンタ３にセットされた後、そのアドレス“１００１”の命令＃１が命令キャッシュ１から命令レジスタ２にフェッチされる。

【００９４】このように、ＣＡＬＬ命令の飛び先にＧＥＴＡ（１，ｒ）を挿入し、当該ＧＥＴＡ（１，ｒ）命令を利用してＣＡＬＬ命令のアドレスをレジスタファイル７内の指定レジスタ（ｒレジスタ）に取出すようにすると共に、手続き終了位置に、当該レジスタの示すアドレスの次のアドレスにリターンするためのＲＥＴ命令を挿入することで、戻りアドレスをリンクレジスタに設定する操作を行わなくても、ＲＥＴ命令により手続き呼出しから戻るときに、ＣＡＬＬ命令のアドレスの次のアドレス（手続き呼出しの戻りアドレス）に実行を移すことができる。

【００９５】なお、前記実施例では、書込み先ポインタ４２がデクリメントされるものとして説明したが、インクリメントされるものであっても構わない。但し、この場合には加算器４３に代えて減算器を設け、ＧＥＴＩ（ｉ，ｒ）またはＧＥＴＡ（ｉ，ｒ）命令の実行時に、この減算器により書込み先ポインタ４２の値からＧＥＴＩ（ｉ，ｒ）またはＧＥＴＡ（ｉ，ｒ）命令中のｉフィールドの値「ｉ」を減算することで、取出しの対象となる循環バッファ４１内エントリを指すエントリアドレスを求める必要がある。

【００９６】また、前記実施例では、命令キャッシュ１から命令レジスタ２にフェッチされた命令が次のサイクルでデコードされるものとして説明したが、処理の高速化のために、命令レジスタ２に代えて命令バッファを設けて命令キャッシュ１から命令バッファへの命令の先取りを行うようにしても構わない。但し、この場合には、命令キャッシュ１から命令バッファにフェッチされた命令がデコードされるまで、何サイクルかの遅延が生じるため、この遅延に同期して、書込み先ポインタ４２の内容を遅延させて対応する命令がデコードされるまで保持するための（パイプラインレジスタ群等の）機構が必要となる。そして、ＧＥＴＩ（ｉ，ｒ）またはＧＥＴＡ（ｉ，ｒ）命令の実行では、同命令中のｉフィールドの値との演算に、書込み先ポインタ４２の値ではなくて、当該機構に保持されている現在デコード中の命令が記録されている循環バッファ４１内エントリを指すアドレスを用いるようにする。

【００９７】また、前記実施例では、循環バッファ４１には命令およびアドレスの対を格納するものとして説明したが、ＧＥＴＡ命令を利用しない演算処理装置であるならば命令だけを、ＧＥＴＩ命令を利用しない演算処理装置であるならばアドレスだけを格納するようにしても構わない。

【００９８】

【発明の効果】以上詳述したようにこの発明によれば、次に列挙する効果を得ることができる。

（１）指定命令数ｎｏｐ化命令（ＳＫＩＰ（Ｎ）命令）で指定された飛ばしたい個数だけ後続の命令をｎｏｐ化することができるため、分岐なしで条件処理を実現することができる。

（２）履歴を取得したい命令よりｉサイクル後に実行される位置に命令履歴取出し命令（ＧＥＴＩ（ｉ，ｒ）命令）を用意しておくことで、当該取出し命令が実行された場合に、当該取出し命令で指定されたｉサイクル前の命令を命令履歴としてバッファ手段から簡単に取出すことができる。

（３）指定命令数ｎｏｐ化命令（ＳＫＩＰ（Ｎ）命令）によりｎｏｐ化される命令部分に、即値データ（即値オペランドデータ）または手続き呼出しの固定引数を設定しておき、更にその後に命令履歴取出し命令（ＧＥＴＩ（ｉ，ｒ）命令）を用意して、これらを実行させることにより、当該取出し命令で指定されたｉサイクル前の命令部分の即値データまたは固定引数をバッファ手段から簡単に得ることができる。このように、即値データまたは引数をレジスタからロードして得る必要がないため、レジスタからのロードによるメモリのオーバヘッド、お

よびレジスタを使用することによるコストの増加を防止することができる。

（４）実行開始サイクルよりＮサイクル後にエラートラップの発生する可能性のある命令（対象命令）が実際にエラートラップした場合の飛び先に、サイクル数Ｎ＋１を指定するアドレス履歴取出し命令（ＧＥＴＡ命令）を用意しておくことで、上記対象命令がエラートラップして当該取出し命令に実行が移った場合に、当該取出し命令で指定されたＮ＋１サイクル前の命令のアドレス、即ち上記対象命令のアドレスをエラー発生命令のアドレスとしてバッファ手段から簡単に取出すことができる。

（５）手続き呼出し命令の飛び先に、サイクル数１を指定するアドレス履歴取出し命令（ＧＥＴＡ（ｉ，ｒ）命令）を用意しておくことで、当該取出し命令が呼出された場合に、当該取出し命令で指定された１サイクル前の命令のアドレス、即ち上記手続き呼出し命令のアドレスを、戻りアドレスの１つ前のアドレスとしてバッファ手段から簡単に取出すことができる。したがって、従来のように、戻りアドレスをリンクレジスタに設定しておく必要がなくなる。

【図面の簡単な説明】

【図１】この発明の一実施例に係る演算処理装置の主要部の構成を示すブロック図。

【図２】同実施例で適用される新規な命令のフォーマット図。

【図３】同実施例における処理の流れを図２に示した命令がフェッチされた際の処理を中心に説明するためのフローチャート。

【図４】同実施例におけるバッファ機構４の機能を説明するための図。

【図５】同実施例において指定命令数ｎｏｐ化命令（ＳＫＩＰ命令）を実行することにより実現される指定命令数ｎｏｐ化機能を説明するための図。

【図６】同実施例において命令履歴取出し命令（ＧＥＴＩ命令）を実行することにより実現される指定サイクル前の命令の取出し機能を説明するための図。

【図７】同実施例においてアドレス履歴取出し命令（ＧＥＴＡ命令）を実行することにより実現される指定サイクル前の命令のアドレスの取出し機能を説明するための図。

【図８】同実施例において分岐なしの条件処理を実現するための命令列（モジュール）を、当該条件処理を記述したソースコード、および従来の分岐ありの条件処理となる命令列（モジュール）と対比させて示すもので、図８（ａ）はソースコード、図８（ｂ）は当該ソースコードのコンパイル結果である従来の命令列、図８（ｃ）当該ソースコードのコンパイル結果である同実施例における命令列。

【図９】同実施例においてＧＥＴＩ命令を利用して、指定サイクル前の命令を取得する命令履歴取出しを説明するための図。

【図１０】同実施例においてＳＫＩＰ命令でｎｏｐ化された命令部分に即値データを格納しておき、当該即値データをＧＥＴＩ命令を利用してレジスタに取出す即値オペランド指定を説明するための図。

【図１１】同実施例においてＳＫＩＰ命令でｎｏｐ化された命令部分に引数を格納しておき、当該引数をＧＥＴＩ命令を利用してレジスタに取出す固定引数引渡しを説明するための図。

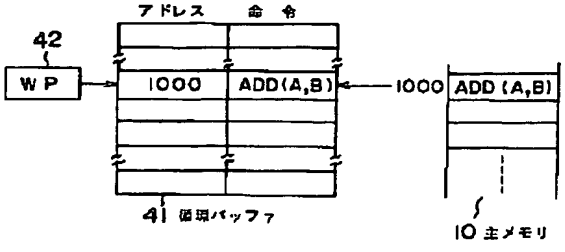
【図１２】同実施例においてエラートラップが発生したらＧＥＴＡ命令に飛ぶようにしておき、当該ＧＥＴＡ命令を利用してエラー発生命令のアドレスをレジスタに取出すエラー発生命令のアドレス取出しを説明するための図。

【図１３】同実施例においてＣＡＬＬ命令（手続き呼出し命令）の飛び先にＧＥＴＡ命令を挿入すると共に、手続き終了位置にＲＥＴ命令（リターン命令）を挿入しておき、ＧＥＴＡ命令を利用してＣＡＬＬ命令のアドレスをレジスタに取出して戻りアドレスの指定に用いるようにした手続き呼出し時の戻りアドレス指定を説明するための図。

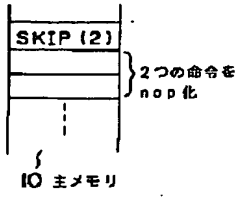
【符号の説明】

１…命令キャッシュ、２…命令レジスタ、３…プログラムカウンタ（ＰＣ）、４…バッファ機構、５，６…セレクタ、７…レジスタファイル、８…命令デコーダ、９…命令ｎｏｐ化機構、１０…主メモリ、４１…循環バッファ、４２…書込み先ポインタ（ＷＰ）、４３…加算器（ＡＤＤ）、８１…セット信号、８２…選択制御信号、９１…命令ｎｏｐ化制御信号。

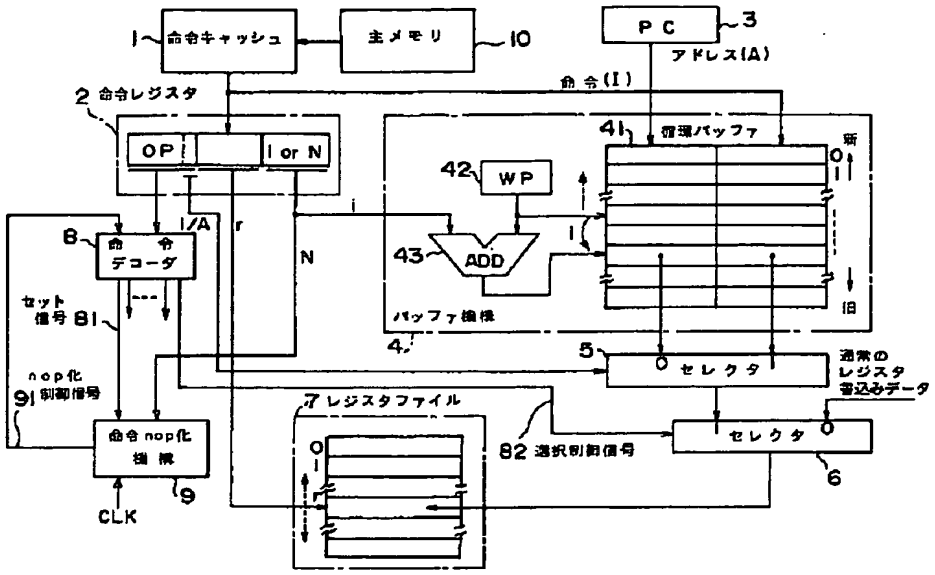
【図４】



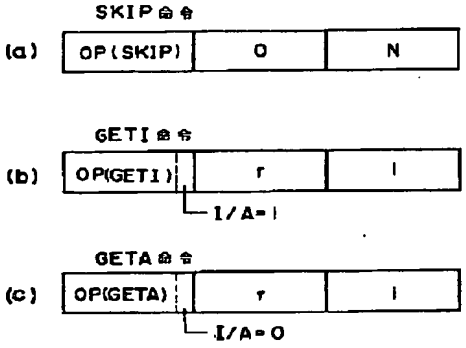
【図５】



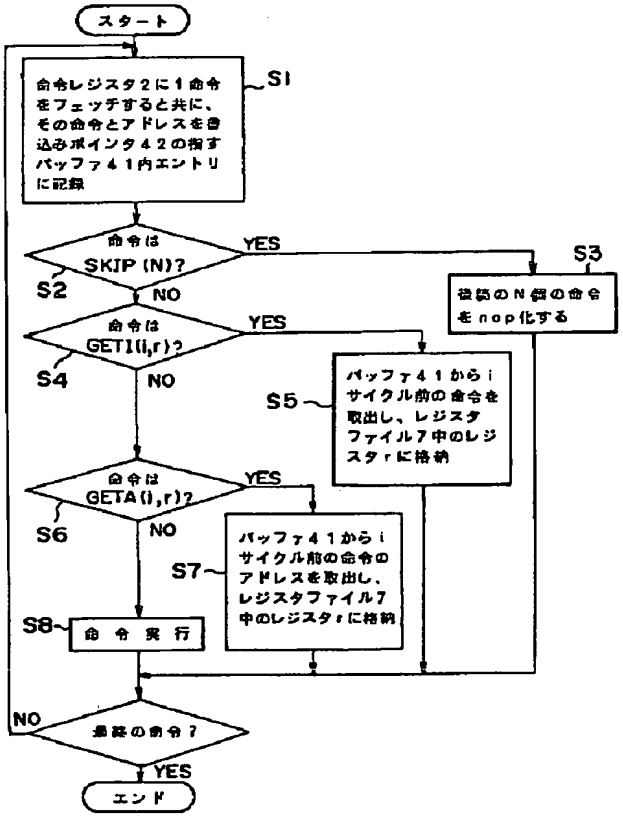
【図１】


命令キャッシュ
主メモリ
10
PC
3
アドレス(A)
命令(I)
2 命令レジスタ
OP
I or N
41 循環バッファ
新
旧
42 WP
43 ADD
I/A
r
N
8 命令デコーダ
セット信号 81
nop化制御信号 91
バッファ機構
4
5 セレクタ
通常のレジスタ書込みデータ
7 レジスタファイル
82 選択制御信号
セレクタ
6
命令nop化機構
9
CLK

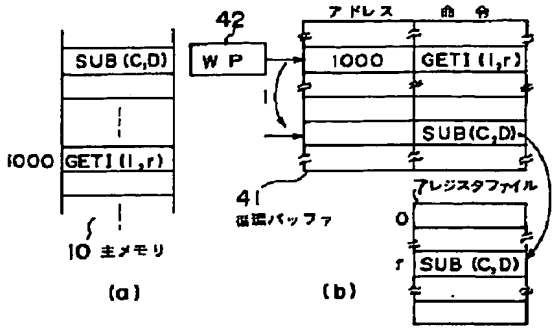
【図２】


SKIP命令
(a)
OP(SKIP)
0
N
GETI命令
(b)
OP(GETI)
r
I
I/A=I
GETA命令
(c)
OP(GETA)
r
I
I/A=0

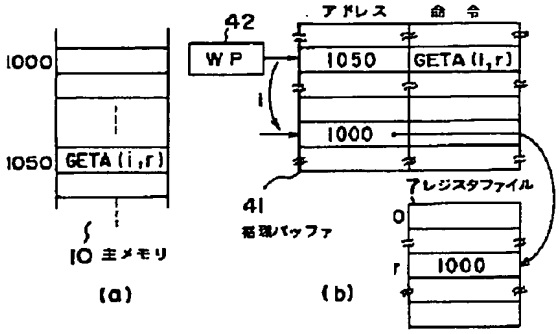
【図３】


スタート
命令レジスタ2に1命令をフェッチすると共に、その命令とアドレスを書込みポインタ42の指すバッファ41内エントリに記録
S1
命令はSKIP(N)?
YES
NO
S2
S3
後続のN個の命令をnop化する
命令はGETI(I,r)?
YES
NO
S4
バッファ41からiサイクル前の命令を取出し、レジスタファイル7中のレジスタrに格納
S5
命令はGETA(I,r)?
YES
NO
S6
バッファ41からiサイクル前の命令のアドレスを取出し、レジスタファイル7中のレジスタrに格納
S7
命令実行
S8
最後の命令?
NO
YES
エンド

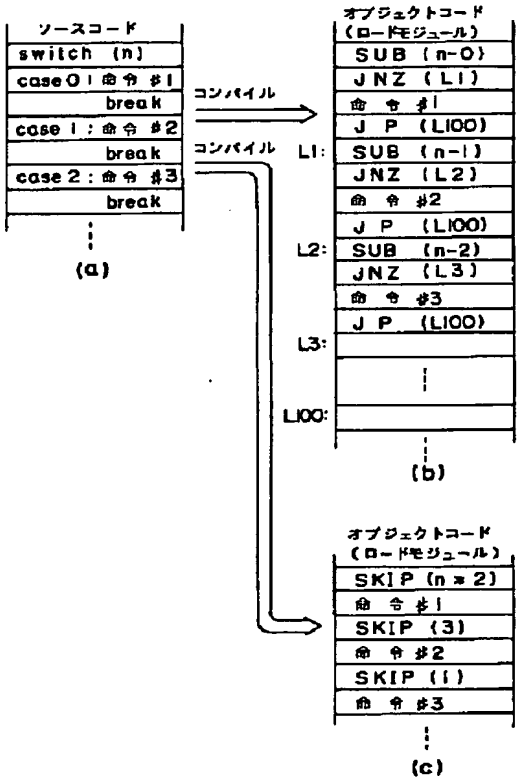
【図6】



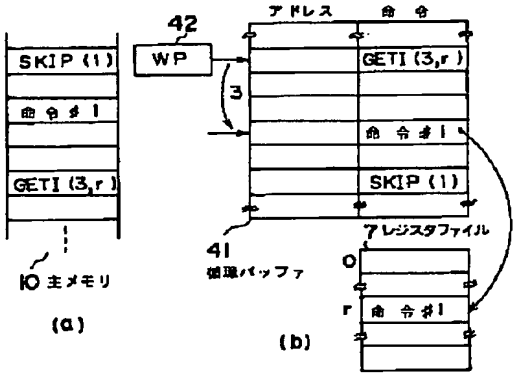
【図7】



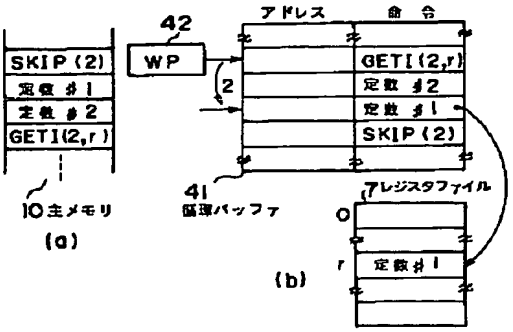
【図8】



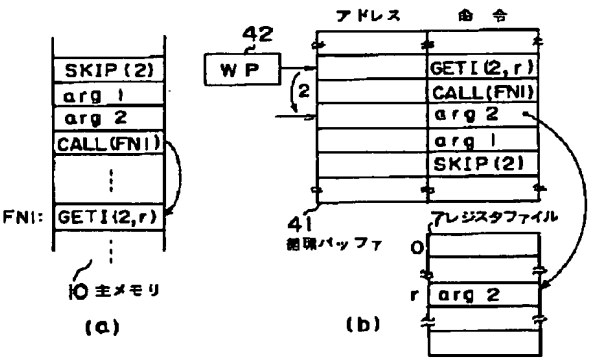
【図9】



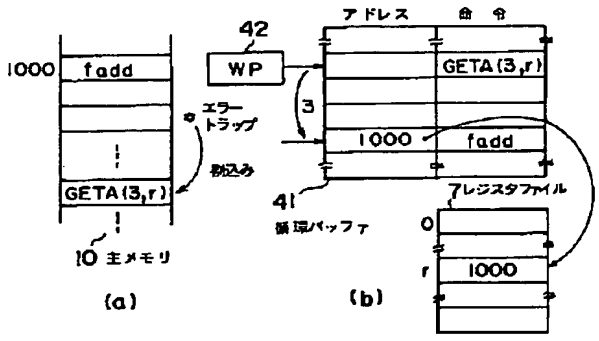
【図10】



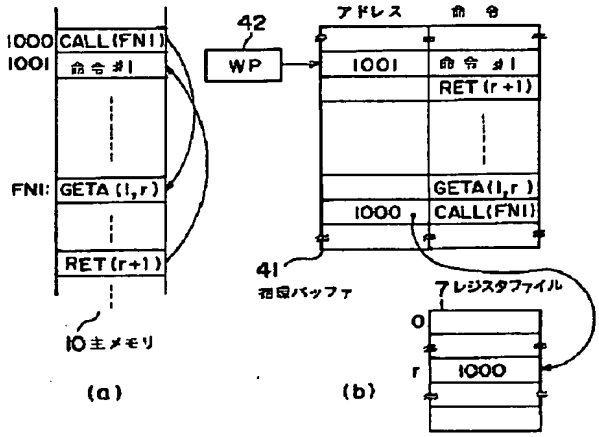
【図11】



【図１２】



【図１３】



フロントページの続き

(72)発明者　森　良哉
東京都府中市東芝町１番地　株式会社東芝府中工場内

(72)発明者　武内　和昭
東京都府中市東芝町１番地　株式会社東芝府中工場内